Java: String? ??? (Immutability)

---

1. ??

Java의 `String` 클래스는 불변 객체**(Immutable Object)** 로 설계되어 있다 .

- 즉 , 한 번 생성된 문자열은 절대로 수정되지 않으며 , 문자열을 변경하려고 시도하면 항상 새로운 객체가 생성된다 .
- 이러한 불변성은 보안, 성능, 스레드 안정성 등 다양한 이유로 매우 중요한 특징이다 .

---

2. ? ??(Immutable)???

불변성은 단순히 설계 철학이 아니라 , 실제 Java 플랫폼의 안정성과 효율성을 위한 필수적인 요소이다 .

2.1 ??(Security)

- 문자열은 종종 파일 경로 , 데이터베이스 연결 문자열 , 네트워크 주소 등에 사용됨 .
- 가변 문자열이라면 악의적인 코드가 문자열을 수정해 보안 구멍을 만들 수 있음 .

```
String dbUrl = "jdbc:mysql://localhost";
dbUrl.replace("localhost", "hacker-site.com");  // String은 수정되지 않음
```

2.2 ??(Performance)

- **String Constant Pool**을 통해 동일한 문자열을 재사용할 수 있음 .
- 이 기능이 제대로 작동하려면 문자열이 절대 바뀌지 않아야 함 .

```
String a = "hello";
String b = "hello";
System.out.println(a == b);  // true → 동일 객체 공유
```

2.3 ??? ???(Thread Safety)

- 불변 객체는 내부 상태가 변경되지 않기 때문에 , 동기화 없이도 안전하게 공유 가능 하다 .
- 멀티스레드 환경에서 추가 비용 없이 안정적인 동작 보장 .

---

3. ?? ?? – ??? ??? “? ?? ??”

3.1 ?? ??

```
String a = "hello";
String b = a;

a = a + " world";

System.out.println(a);  // hello world
System.out.println(b);  // hello
```

- `"hello"`는 리터럴 풀에 존재
- `"hello world"`는 Heap에 새로운 객체로 생성
- 변수 b는 여전히 `"hello"`를 참조

3.2 ??? ?? ??

```
[초기 상태]
a ──▶ "hello"
b ──┘

[수정 후]
a ──▶ "hello world"
b ──▶ "hello"
```

□□ 변경이 아닌 “대체”다. 기존 객체는 절대 바뀌지 않음 !

---

4. ?? ??? ?? ??

항목	설명
변경 가능 여부	□ □□□ – 항상 새로운 객체 생성됨

스레드 안전성	□ □□□ 없이 공유 가능
공유 가능 여부	□ □□□ 리터럴 풀에서 객체 공유 가능
메모리 구조	Heap 영역 + String Constant Pool 최적화
관련 API 특징	`replace()`, `concat()` 등 모두 새 객체 반환

---

5. ?? ??? ??

5.1 ?? ?? (Immutable)

- `String`
- `Integer`
- `Boolean`
- `LocalDate`, `LocalTime`, `BigDecimal`

5.2 ?? ?? (Mutable)

- `StringBuilder`
- `StringBuffer`
- `ArrayList`
- `HashMap`

□□ 불변 클래스는 **equals(), hashCode()** 재정의에 유리하며, 객체 캐싱 및 보안 측면에서도 장점을 가짐.

---

6. ???? ???

6.1 ?? ?? ? ?? ??

```
String s = "";
for (int i = 0; i < 1000; i++) {
    s += "a";  // 매 반복마다 새 객체 생성
}
```

→ 이 방식은 **1000**개의 **String** 객체를 생성함
→ 해결 방법: `StringBuilder` 사용

```
StringBuilder sb = new StringBuilder();
for (int i = 0; i < 1000; i++) {
    sb.append("a");
}
String result = sb.toString();
```

---

7. ??

- `String`의 불변성은 Java 언어의 안정성과 성능 최적화를 위한 중요한 설계 원칙이다.
- 문자열이 자주 변경된다면 `StringBuilder`나 `StringBuffer`를 적극적으로 활용해야 한다.
- 불변 객체는 보안성, 공유성, 스레드 안전성 측면에서 뛰어난 장점을 가진다.

Revision #6
Created 20 May 2025 02:21:03 by Dain
Updated 20 May 2025 06:22:51 by Dain